Problem Solving by Computer: Project 1

March 3, 2020

1 Cubic Taxicab Number

1.1 Description

The task for this question was to find and return the closest cubic taxicab number N (integer) larger or equal to a given input. A cubic taxicab number is the result of adding together two unique pairs of perfect cubes.

An example of a cubic taxicab number is 1729. This number can be expressed as a sum of two pairs of cubes: $1^3 + 12^3$ and $9^3 + 10^3$.

1.2 Methodology

Given an integer N, we want to increment N until we find a taxicab number as defined above. My method would start with fixing N as the input, then trying to find two pair of cubes and incrementing N by 1 if N is not determined to be a taxicab number. We can first cube each integer k and j, such that $1 \leq k, j < N$ and add each result of k and j together, where $k \neq j$. Until we find two pairs which add up to the closest N increment. We can call this the 'brute force' approach.

However, it is possible to reduce the integers which k and j can possibly equal by finding a suitable ceiling. In this problem, such a ceiling can be found by calculating $\sqrt[3]{N}$ and rounding down using the floor function in Matlab. We can see see this is true because if we let $k = \sqrt[3]{N}$ and let $j = 1$ (the smallest positive integer), $k^3 + j^3 > N$ for all N.

Further, we want to improve the approach of iterating k and j. In this step, we can let $j = 1$ and $k = \text{floor}(\sqrt[3]{N})$. Our aim is to find $k + j = x$ such that $x = N$. To optimise this, we can set conditions of stepping through each k and j. If the resulting x is greater than N, then we know we have to reduce the sum, which can only be done by reducing the k, therefore we reduce k by 1. Conversely, if x is less than N, we have to increase j by 1. If x is equal to N, then we increase j by one and reduce k by one as we know we have found a pair which equates to N. We only reduce or increase by 1 to ensure we test all positive integers. Once $k = j$, we know we have tested all integers, and if we haven't found two pairs of cubes which equate to N, we have verified it is not a taxicab number.

1.3 Algorithm

```
function ctn = CubicTaxicabNum(N)
% CUBICTAXICABNUM return the smallest cubic taxicab number greater than
% or equal to N
count = 0;
N = N - 1;

while(count < 2)
    count = 0;

    UB = floor(nthroot(N, 3));

    i1 = 1; % Set the left iterate as the lower bound

    i2 = UB; % Set the right iterate as the upper bound

    while i1 <= i2 % Check to see if we have tested all values yet

        cubed = i1^3 + i2^3;

        if (cubed == N) % If current sum is the N, we look for the next
                        % pair of cubes

            i1 = i1 + 1;
            i2 = i2 - 1;
            count = count + 1;

        elseif (cubed < N) % If current sum is too small, we move the left
                           % iterate

            i1 = i1 + 1;

        elseif (cubed > N) % If current sum is too small, we move the
                           % right iterate

            i2 = i2 - 1;
        end
    end

    ctn = N;   % Set the output as current N
    N = N + 1; % Iterate to the next N

end
% The program will end once we have found the closest taxicab number as the
% the count of pair of cubes will be two, under the assumption there is infinite
% amount of taxicab numbers.
end
```

1.4 Analysis

In our task, we were given the input $i = 36032$ and we were to find the closest taxicab number to this integer. Using my algorithm wecan verify that the closest taxicab number larger than the input is $N = 39312$.

The 'brute force' approach tries all integers less than N with each other, therefore this approach has a time complexity of $O(n^2)$. While this is not computationally expensive for a modern computer for the given input, the optimisations implemented reduce the time complexity to $O(\sqrt[3]{n})$ as all the computations in the loop are being done in constant time. This is important to optimise in the case, where we test multiple numbers or very large numbers.

Fermat's Last theorem states, there is no such integer $n > 2$, such that $a^n + b^n = c^n$ for any three positive integers a, b and c. We can implement this easily and check if any N is a cubed number, and if so, we don't check the iterates for it. This reduces the number of calculations checked further.

2 Catalan's Constant Approximation

2.1 Description

This task was to find two positive integers p and q, such that p/q is the best approximation to Catalan's constant G, under the constraint, $p + q$ is less than a positive integer N.

Catalan's constant is the Dirichlet beta function at $\beta(2)$ and can be defined as

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)^k} \approx 0.915965594177219...$$

For the purpose of this task, we will assume $G = 0.915965594177219$ and not evaluating G further as it is unknown if it is irrational or transcendental.

2.2 Methodology

In this task, we have two conditions: $p/q \approx G$ and $p + q \leqslant N$. Using these two conditions, we can optimise the values we try up to N.

Assume $p/q = G$, then $p = Gq$. Observe $Gq + q \leqslant N$. From this we can derive the upper bound of q as the truncated value calculated from $q \leqslant N/(G+1)$. Following this, we can also approximate p easily by calculating $p = Gq$ and rounding the output to the nearest positive integer.

Using these calculations, and iterating q from 1 to the upper bound, we can find the best approximation under the constraint. We iterate q and calculate the absolute difference from G and once the minimum difference from G has been found, we know we have found the best approximation.

2.3 Algorithm

```
function [p, q] = RatAppCat(N)
% RATAPPCAT The best rational approximation p/q of the Catalan 's constant ,
% among all pairs of (p,q) such that p+q <=N

G = 0.915965594177219; % Define Catalan's constant

qMax = floor(N/(G+1)); % Calculate upper bound of q
p = 0;
q = 0;
bestApprox = 1; % Variable for best approximation

for i = 1:qMax % The iterate for the q value

    j = round(i * G); % The iterate for the p value

    currentApprox = abs(j/i-G); % Find the current error

    if currentApprox < bestApprox % If current error is better than the
                                  % previous error, set it as the best
                                  % approximation and save the best p and q

        bestApprox = currentApprox;
        p = j;
        q = i;
    end
end
```

2.4 Analysis

The task was to find the best approximation of G under the constraint $p + q$ be equal or less than $N = 2018$ and $p/q \approx G$. Using the algorithm as described above, the best approximation is calculated when $p = 109$ and $q = 119$.

As with the problem above, the time complexity would be $O(n^2)$ if we were to test each p and q up to N. However, this is not so expensive as it is simply a division function. Nevertheless, our optimisations of finding the bounds for p and q reduced this to a problem of time complexity $O(n)$.

3 Sum of Reciprocal Squares with Prime Factors

3.1 Description

The Basel problem $\sum_{k=1}^{\infty} \frac{1}{k^2}$ was shown to be exactly equal to $\pi^2/6$. Given a similar series, we want to truncate a certain number of terms and produce an answer to a reasonable degree of accuracy.

The series we are given is the Dirichlet series for the Liouville function defined as

$$S = \sum_{k=1}^{\infty} \frac{(-1)^{\Omega(k)}}{k^2}$$

where $\Omega(k)$ is function counting the number of prime factors of k with multiplicity. For instance, $36 = 2^2 \cdot 3^2$, therefore $\Omega(36) = 4$.

The series is related to the Riemann zeta function as $S = \zeta(4)/\zeta(2) = \pi^2/15$, however this result will not be used in the calculations as the problem is to find an algorithm for the best approximation and not the concrete answer.

3.2 Methodology

The terms in the series S are positive or negative dependent on $\Omega(k)$, also known as the prime omega function. This function has no discernible pattern. However, after a sufficient number of terms, the term being positive or negative is close enough to a coin flip [1].

The direct comparison test is described as, if $\sum_{n=1}^{\infty} a_n$ is absolutely convergent and $|b_n| \leq |a_n|$ for all $n \in \mathbb{N}$ then $\sum_{n=1}^{\infty} b_n$ is absolutely convergent. Using the Basel problem and S, we can say S is absolutely convergent.

Due to the series S being convergent, after a certain amount of terms, we will reach an approximate answer to some degree of accuracy. In my method, wewill calculate an upper bound and a lower bound for multiple windows of the series and the difference between the bounds will be my degree of accuracy. The bounds will be calculated as the maximum and minimum values of the series in that window.

We could also use the alternating series estimation theorem[2], however the terms to not alternate with any pattern therefore we need to calculate a large amount of terms before we can use the theorem. [1]

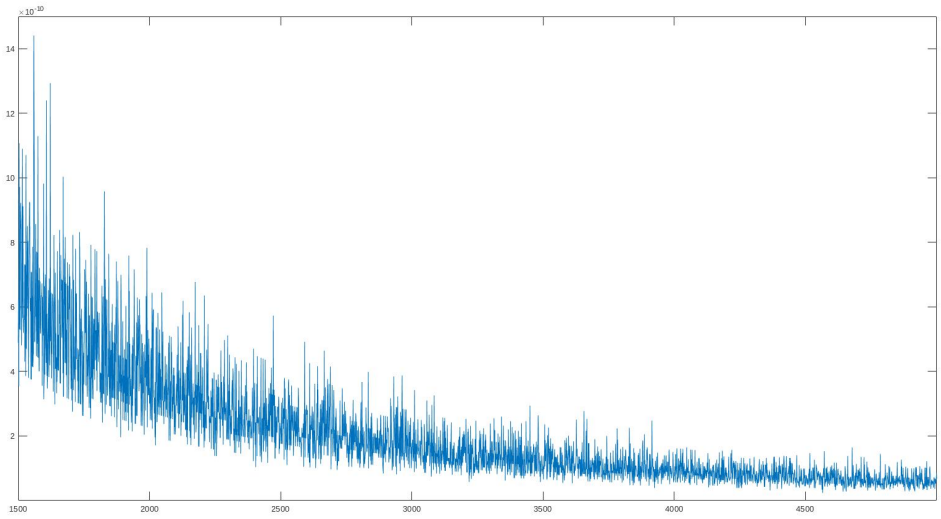
3.3 Algorithm

```
function SumPF

format long
window = 100; % Size of each group of terms
upper = 500000; % Number of terms tested
min = ones(1, upper/window);
max = zeros(1, upper/window);
errorRate = zeros(1, upper/window);
sum = 1;
k = 1;


for n = 1:window:upper

    for j = 1:window
        k = k + 1;
        next_k = (-1)^length(factor(k)) / k^2; % Working out the next term
        sum = sum + next_k;

        if (sum > max(1, ceil(n/window))) % Finding the maximum sum in the
                                          % in the window
            max(1, ceil(n/window)) = sum;
        end


        if (sum < min(1, ceil(n/window)))% Finding the minimum sum in the
                                          % in the window

            min(1, ceil(n/window)) = sum;
        end
    end


end

for x = 1:upper/window
    errorRate(1, x) = max(1, x) - min(1, x); % Finding the error rate
                                             % between the upper bound and
                                             % lower bound
end

plot(5000:(upper/window), errorRate)

end
```

3.4 Analysis

The accuracy of the approximation can be derived from the decreasing difference between the bounds up to the terms specified. For instance, if wewant the accuracy to be less than 10^{-6}, we will calculate up until when the error difference bound is less than the degree of accuracy. In the figure below, you can see the error decreases as the terms are added, as predicted. With 500,000 terms, we can get less than 10^{-12} error difference. This follows since we know the series is convergent.

Using the results for the comparison test, it also possible to bound using the Basel problem as each kth term of the Basel problem will be greater than the kth term of the series S. However, this will require computing each value of the Basel problem and will be nearly as expensive to the computing power as computing each term of the series S.



References

[1] Eswaran, K.. (2017). The Dirichlet Series for the Liouville Function and the Riemann Hypothesis. `https://www.researchgate.net/publication/312492134_The_Dirichlet_Series_for_the_Liouville_Function_and_the_Riemann_Hypothesis`

[2] Villarino, Mark B.. (2018) The Error in an Alternating Series `https://www.tandfonline.com/doi/full/10.1080/00029890.2017.1416875`